Introduction à la programmation
Travaux pratiques: séance 2
INFO0201-1

X. Baumans
(xavier.baumans@ulg.ac.be)

Programme de la séance

- Rappels déclaration de variable, opérateurs, entrée / sortie

Programme de la séance

- Rappels déclaration de variable, opérateurs, entrée / sortie
- Expression booléenne

Programme de la séance

- Rappels déclaration de variable, opérateurs, entrée / sortie
- Expression booléenne
- Structure de contrôle "if...else"

Rappels : déclaration, initialisation, affectation, échange

Déclaration, initialisation, affectation :

```
1 int a; //déclaration
2 a = 3; // initialisation (affectation initiale)
3 double b = 2.5; // déclaration et initialisation
4 b = 5.98; // affectation d'une nouvelle valeur à b
```

Échange (utilisation d'une variable tampon) :

```
1 float a = 3.2; float b = 6.8; float t;
2 t = a; //on stocke la valeur de a dans t (t = 3.2)
3 a = b; //on affecte la valeur de b à a (--> a = 6.8)
4 b = t; //on affecte l'ancienne valeur de a (stockée
5        //dans t) à b (--> b = 3.2)
```

Rappels : opérateurs

```
1  int a = 3;
2  int b = 2;
3  int r;
4
5  r = a+b; // la valeur de l'expression "a+b" est
6           // stockée dans r (--> r = 5)
7
8  r = a*b; // (--> r = 6)
9
10 r = a-b; // (--> r = 1)
11
12 r = a/b; // (--> r = 1 !!division entière!!)
13
14 r = a%b; // (--> r = 1 reste de la division entière)
```

Rappels : opérateurs

Échange en utilisant que les opérateurs arithmétique de base (pas de tampon) :

```
1  int a = 3;
2  int b = 2;
3
4  //Exemple avec l'addition
5
6  // --> a = 3 & b = 2
7
8  a = a+b; // --> a = 5 & b = 2
9
10 b = a-b; // --> a = 5 & b = 3
11
12 a = a-b; // --> a = 2 & b = 3
```

Rappels : entrée / sortie

```
1  int a;
2  double b;
3
4  cout<<"Entrez une valeur entière: ";
5  cin>>a;
6
7  cout<<"Entrez une valeur réelle: "; cin>>b;
8
9  cout<<"Leur produit vaut: " << a*b ;
10
11 cout<<"Leur somme vaut: " << a+b ;
```

Inutile de déclarer des variables pour stocker la somme et le produit de a et b (à moins qu'on désire utiliser ces valeurs dans la suite !)

Expression booléenne

Expression booléenne

Une expression booléenne est une expression dont l'évaluation peut uniquement résulter en deux valeurs : **true** ou **false**.

Pour des raisons historiques, les booléens peuvent être convertis en entiers et inversément. Les conversions sont définies par les règles suivantes :

- **true** $\rightarrow$ 1
- **false** $\rightarrow$ 0
- Tout entier non nul $\rightarrow$ **true**
- L'entier 0 $\rightarrow$ **false**

```
1 bool a = 0; --> false
2 bool b = 1; --> true
3 bool c = 5 + 2; --> true
```

Expression booléenne

Exemples d'expressions booléennes :

```
1 int a = 5; int b = 10;
2
3 (a==b); // expression évaluée à "false"
4 (a!=b); // expression évaluée à "true"
5 (a<b); // --> true
```

Affectation à un booléen :

```
1 int a = 5; int b = 10; bool c;
2 //L'expression booléenne sera d'abord évaluée
3 //Sa valeur sera ensuite affectée au booléen
4
5 c = (a==b); // la valeur "false" est affectée à c
6 c = (a!=b); // c contient "vraie"
7 c = (30<27); // c est "false"
```

Structures de contrôle : if

Pour construire un programme plus complexe
$\rightarrow$ exécuter certaines instructions sous conditions

Première structure de contrôle : le "**if**"

Sa syntaxe est la suivante : **if**(*condition*) *instruction* ;

condition est une **expression booléenne** qui sera évaluée. Si le résultat est **true**, l'instruction sera exécutée. Si le résultat est **false**, elle sera ignorée.

```
1 int a = 3;
2 cout << "La valeur de a est " << a << endl;
3 if(a > 2)
4   cout << "a est plus grand que 2" << endl;
```

Structures de contrôle : if...else

On peut également exécuter une autre instruction si la condition évaluée n'est pas vérifiée. Cette instruction est introduite par le mot-clé **else** :

```
1 if(a > 2)
2   cout << "a est plus grand que 2" << endl;
3 else
4   cout << "a n'est pas plus grand que 2" << endl;
```

On peut même faire suivre plusieurs conditions :

```
1 if(a > 5)
2   cout << "a est plus grand que 5" << endl;
3 else if(a < 2)
4   cout << "a est plus petit que 2" << endl;
5 else
6   cout << "a est compris entre 2 et 5" << endl;
```

Blocs d'instructions

Dans une structure de contrôle telle que **if...else**, on ne peut exécuter qu'une seule instruction pour chaque condition.

Pour remédier à cette limitation, on utilise des **blocs d'instructions**. Ceux-ci sont constitués de plusieurs instructions placées en parenthèses **{...}** : ils sont considérés comme une seule instruction à exécuter.

```
1 if(a > 5)
2 {
3     cout << "a est plus grand que 5" << endl;
4     int b = 5*2;
5     cout << "5 x a = " << b << endl;
6 }
```

Indentation

Pour améliorer la lisibilité du code source, on aligne les blocs de code en accord avec la structure du programme.

Règle de base : toute section de code qui dépend hiérarchiquement d'une autre est décalée d'une tabulation vers la droite par rapport à celle-ci.

```
1 if(a > 2)
2 {
3     cout << "a est plus grand que 2" << endl;
4     if(a > 4)
5         cout << "a est plus grand que 4" << endl;
6 }
```

Dans le cas d'un bloc d'instruction, on aligne les parenthèses sur l'instruction précédente et on n'indente que le code situé entre les parenthèses.

Exercices

1. Écrire un programme qui affiche la valeur absolue de X (X étant rentré au clavier par l'utilisateur)
2. Tests conditionnels mutliples :
   - Saisir un nombre entier au clavier ;
   - Déterminer si le nombre est divisible par 2 et l'afficher ;
   - Déterminer si le nombre est divisible par 3 et l'afficher ;
   - Déterminer si le nombre est divisible par 5 et l'afficher ;
3. Calculatrice simple :
   - Demander à l'utilisateur quelle opération il souhaite effectuer ;
   - Saisir au clavier les 2 nombres sur lesquels effectuer l'opération ;
   - Afficher le résultat du calcul dans la console.

Exercices

4. Conversion format temporel :
   - Saisir un nombre entier au clavier. Celui-ci correspond à une durée exprimée en secondes ;
   - Afficher à l'écran la durée au format "x heure(s) y minute(s) et z seconde(s)".

   Remarque : n'afficher "heure(s)" et "minute(s)" que si cela est nécessaire.
5. Intersection d'intervalles :
   - Demander à l'utilisateur de saisir 4 nombres A,B,C,D (ordre et valeurs quelconques) ;
   - Lui afficher l'intersection des intervalles [A,B] & [C,D].
6. Résolutions d'équations :
   - Écrire un programme qui résout les équation du premier et du second degré, dont les coefficients sont déterminés par l'utilisateur ;

   Note : Équations de la forme $Ax+b=0$ et $Ax^2+Bx+C=0$, avec A,B,C rentrés au clavier par l'utilisateur. Pour les fonctions mathématiques avancées telles que la racine carrée ($\sqrt{x}$ = sqrt(x)), inclure la bibliothèque <cmath>.